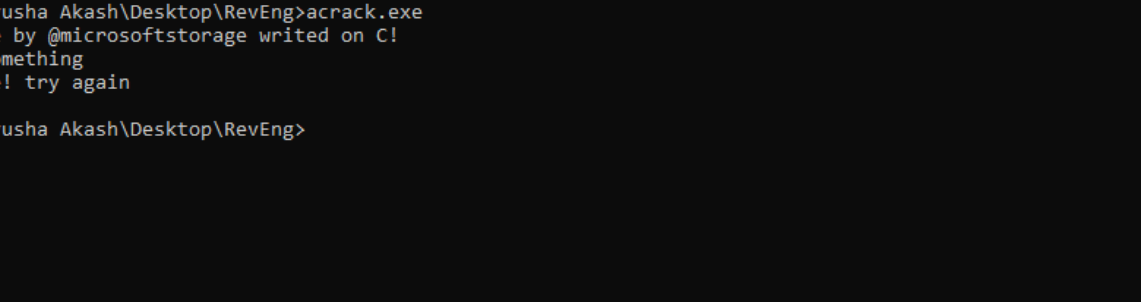


acrack.exe | crackmes.one
EASY LEVEL REVERSE ENGINEERING

So, this a very easy level RE practice for Reverse engineers Based in C. First of all, we want to see what's the application do. So, I just executed it through cmd.



The screenshot shows a Windows command prompt window with the title bar "C:\Windows\System32\cmd.exe". The window contains the following text:

```
C:\Users\Piyusha Akash\Desktop\RevEng>acrack.exe
Mini crackme by @microsoftstorage writed on C!
Nickname: something
bad nickname! try again

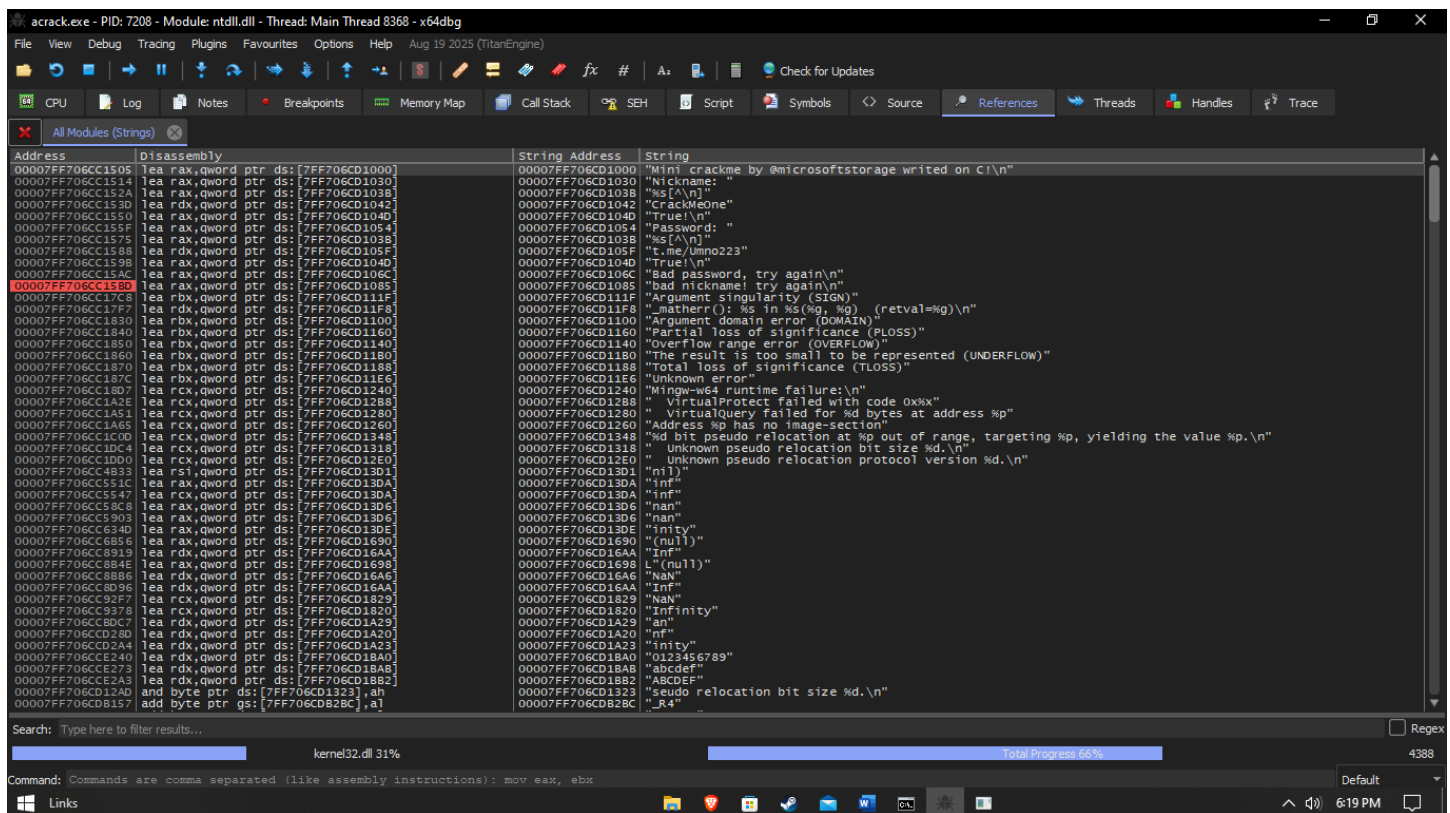
C:\Users\Piyusha Akash\Desktop\RevEng>
```

The text indicates that the user ran the `acrack.exe` program, which is a mini crackme by @microsoftstorage. The user entered the nickname "something", but the program responded with "bad nickname! try again".

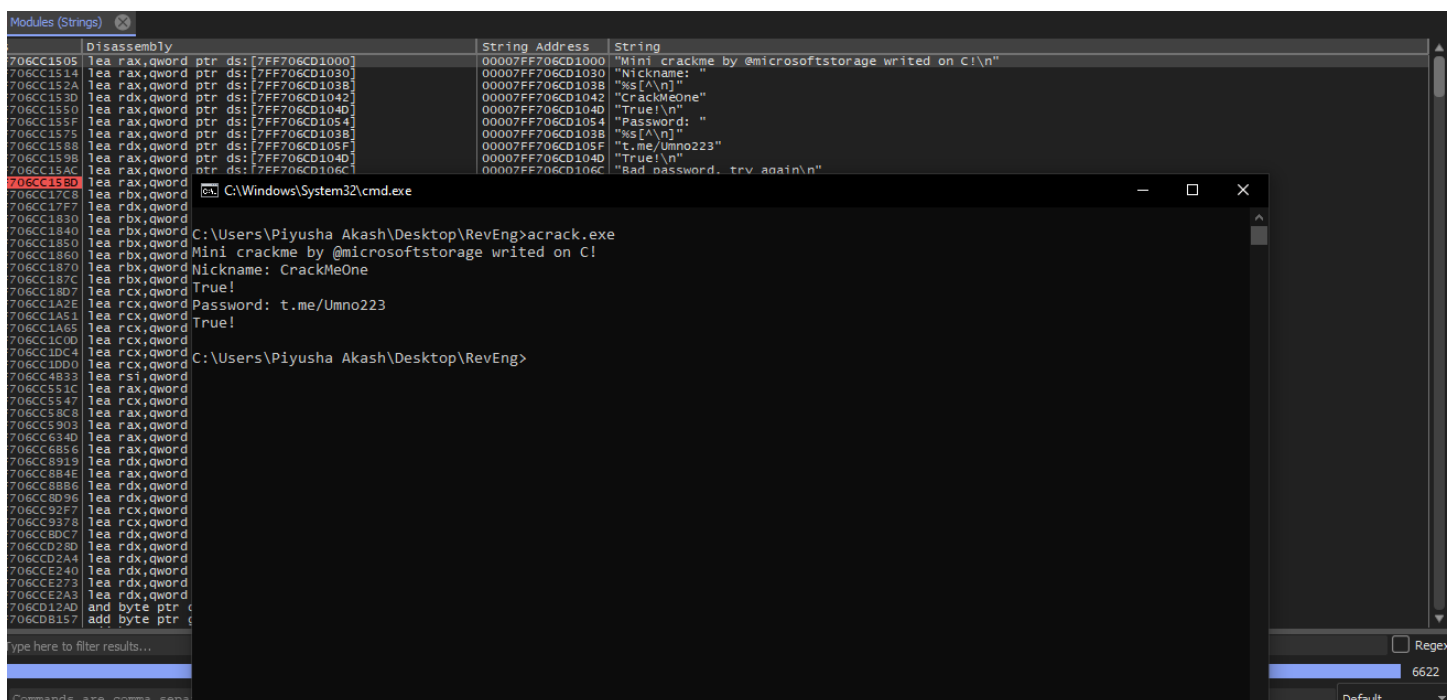
After executing the application, we can see it will ask for Nickname and I just entered "something" as the Nickname and it will give me "bad nickname! Try again." So, we can see it is the wrong Nickname. Let's start Reverse engineer this application. So, in that case I use x64dbg as my tool for reverse this application. I just drag and drop the application to x64dbg.

[illegible]

So, Now I need to looking for strings in this application. So, I will right click the assembly area and go as this. Right click → Search for → All modules → String references. So, I can now see the all strings in this application.



In this section I can see all the string and I will be looking for some hint. And I saw something. I found the password though strings easily. Let's try these credentials.



So, we can see Nickname is CrackMeOne and Password is t.me/Umno223. These credentials are correct and we will get True! Message.

We found the password and nickname but it's not the game. What happen if we can crack the program for return 0 (True! message) for any string. Let's start. So, I just double click the "Nickname" string and then I will go to disassembled section matching to String.

Address	Disassembly	Comment
00007FF706CC1514	48: 8D05 15FB0000	lea rax,qword ptr ds:[7FF706CD1030]
00007FF706CC151B	48: 89C1	mov rcx,rax
00007FF706CC151E	E8 81FFFFFF	call acrack.7FF706CC14A4
00007FF706CC1523	48: 8B45 F8	mov rax,qword ptr ss:[rbp-8]
00007FF706CC1527	48: 89C2	mov rdx,rax
00007FF706CC152A	48: 8D05 0AFB0000	lea rax,qword ptr ds:[7FF706CD103B]
00007FF706CC1531	48: 89C1	mov rcx,rax
00007FF706CC1534	E8 17FFFFFF	call acrack.7FF706CC1450
00007FF706CC1539	48: 8B45 F8	mov rax,qword ptr ss:[rbp-8]
00007FF706CC153D	48: 8D15 FEFA0000	lea rdx,qword ptr ds:[7FF706CD1042]
00007FF706CC1544	48: 89C1	mov rcx,rax
00007FF706CC1547	E8 94DB0000	call <JMP.&strcmp>
00007FF706CC154C	85C0	test eax,eax
00007FF706CC154E	75 6D	jne acrack.7FF706CC158D
00007FF706CC1550	48: 8D05 F6FA0000	lea rax,qword ptr ds:[7FF706CD104D]
00007FF706CC1557	48: 89C1	mov rcx,rax
00007FF706CC155A	E8 45FFFFFF	call acrack.7FF706CC14A4
00007FF706CC155F	48: 8D05 EEFA0000	lea rax,qword ptr ds:[7FF706CD1054]
00007FF706CC1566	48: 89C1	mov rcx,rax
00007FF706CC1569	E8 36FFFFFF	call acrack.7FF706CC14A4
00007FF706CC156E	48: 8B45 F0	mov rax,qword ptr ss:[rbp-10]
00007FF706CC1572	48: 89C2	mov rdx,rax
00007FF706CC1575	48: 8D05 BFFA0000	lea rax,qword ptr ds:[7FF706CD103B]
00007FF706CC157C	48: 89C1	mov rcx,rax
00007FF706CC157F	E8 CCFFFFFF	call acrack.7FF706CC1450
00007FF706CC1584	48: 8B45 F0	mov rax,qword ptr ss:[rbp-10]
00007FF706CC1588	48: 8D15 D0FA0000	lea rdx,qword ptr ds:[7FF706CD105F]
00007FF706CC158F	48: 89C1	mov rcx,rax
00007FF706CC1592	E8 49DB0000	call <JMP.&strcmp>
00007FF706CC1597	85C0	test eax,eax
00007FF706CC1599	75 11	jne acrack.7FF706CC15AC
00007FF706CC159B	48: 8D05 ABFA0000	lea rax,qword ptr ds:[7FF706CD104D]
00007FF706CC15A2	48: 89C1	mov rcx,rax
00007FF706CC15A5	E8 FAFFFFFFFF	call acrack.7FF706CC14A4
00007FF706CC15AA	E8 20	jmp acrack.7FF706CC15CC
00007FF706CC15AC	48: 8D05 B9FA0000	lea rax,qword ptr ds:[7FF706CD106C]
00007FF706CC15B3	48: 89C1	mov rcx,rax

```

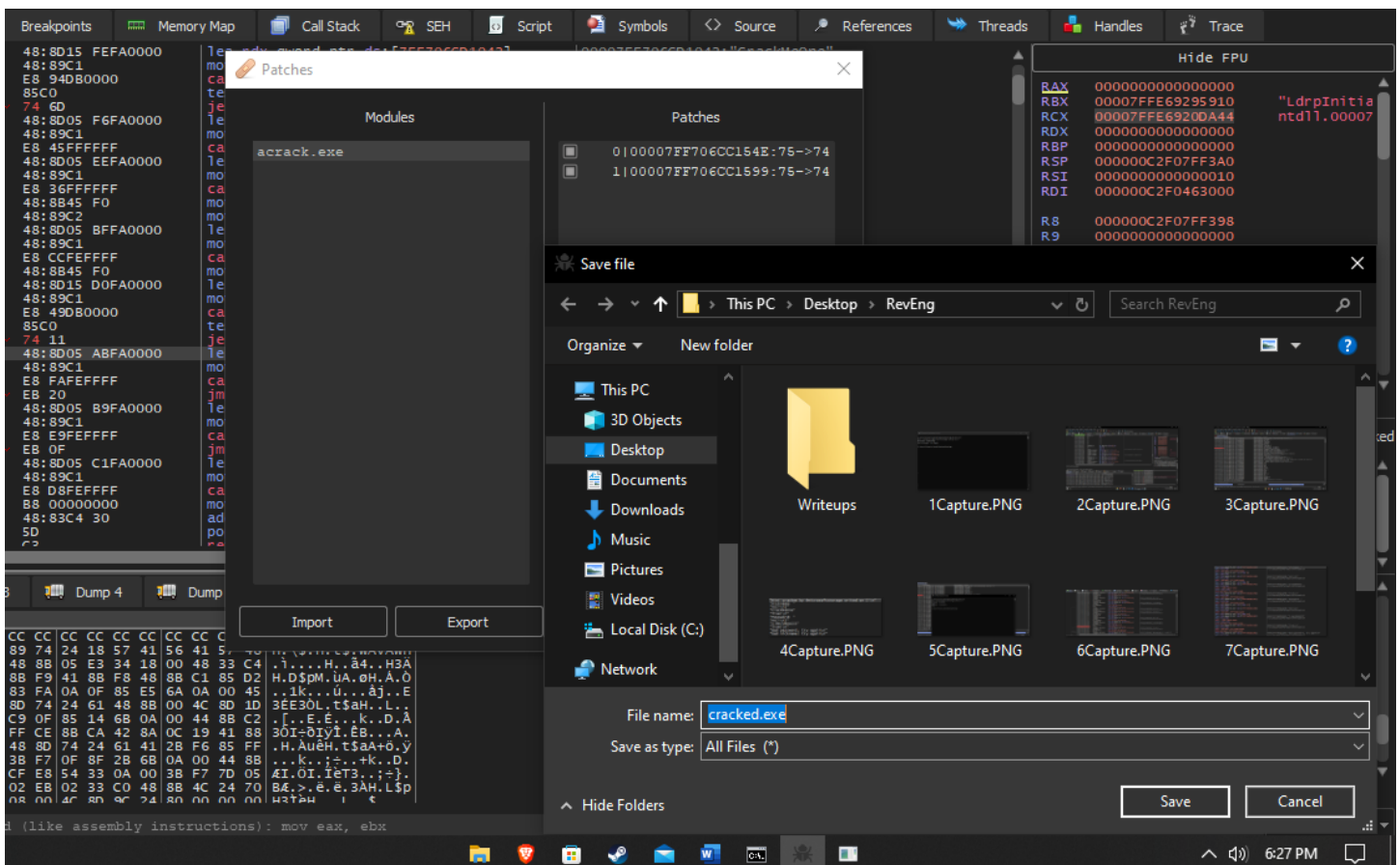
lea rax,qword ptr ds:[7FF706CD1030]
mov rcx,rax
call acrack.7FF706CC14A4
mov rax,qword ptr ss:[rbp-8]
mov rdx,rax
lea rax,qword ptr ds:[7FF706CD103B]
mov rcx,rax
call acrack.7FF706CC1450
mov rax,qword ptr ss:[rbp-8]
mov rdx,qword ptr ds:[7FF706CD1042]
mov rcx,rax
call <JMP.&strcmp>
test eax,eax
jne acrack.7FF706CC158D
lea rax,qword ptr ds:[7FF706CD104D]
mov rcx,rax
call acrack.7FF706CC14A4
lea rax,qword ptr ds:[7FF706CD1054]
mov rcx,rax
call acrack.7FF706CC14A4
mov rax,qword ptr ss:[rbp-10]
mov rdx,rax
lea rax,qword ptr ds:[7FF706CD103B]
mov rcx,rax
call acrack.7FF706CC1450
mov rax,qword ptr ss:[rbp-10]
lea rdx,qword ptr ds:[7FF706CD105F]
mov rcx,rax
call <JMP.&strcmp>
test eax,eax
jne acrack.7FF706CC15AC
lea rax,qword ptr ds:[7FF706CD104D]
mov rcx,rax
call acrack.7FF706CC14A4
jmp acrack.7FF706CC15CC
lea rax,qword ptr ds:[7FF706CD106C]
mov rcx,rax

```

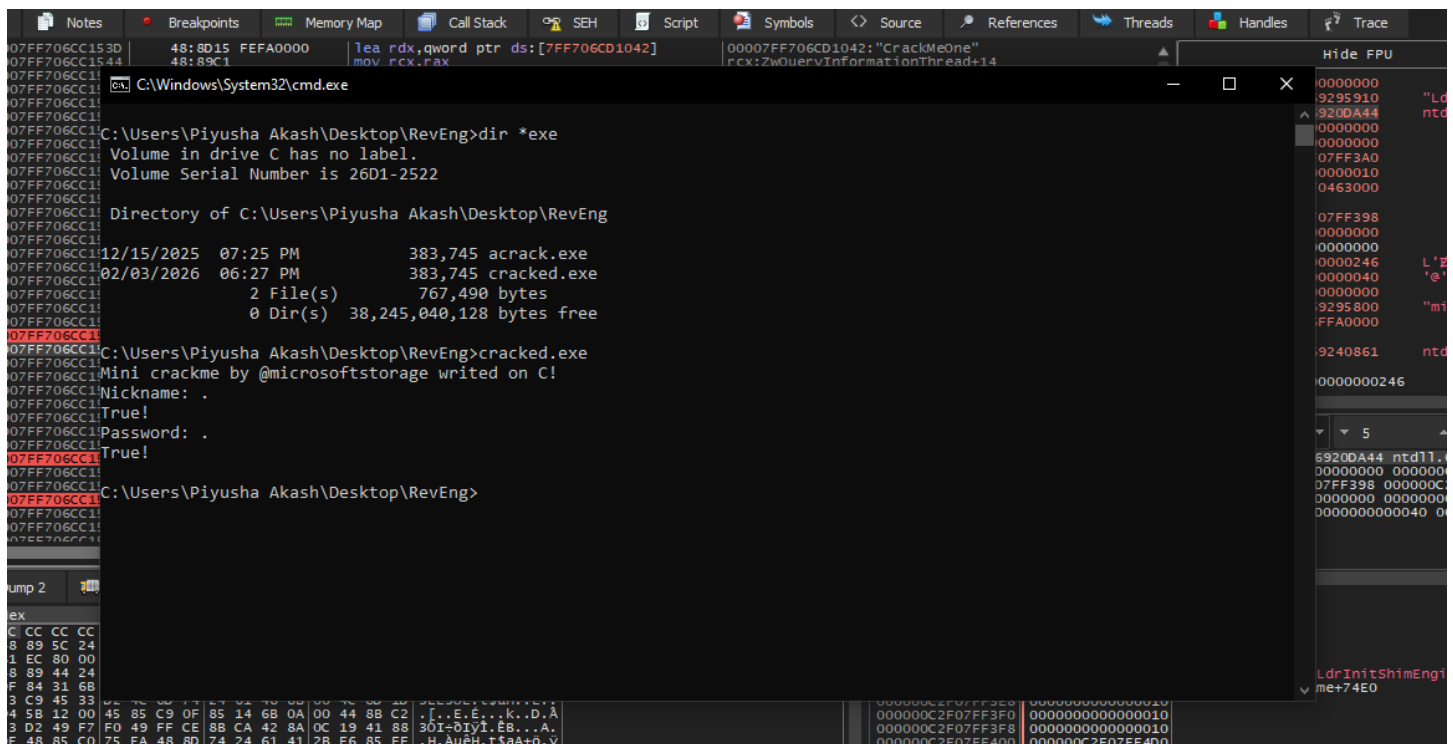
So now I can see Assembly of the program in disassembled section. We need to read this section to understand what's the program do under the hood. So, we can see our password named "CrackMeOne" and in the disassembled section we can see something like "lea rdx, qword ptr ss:[rbp-8]". Under this line we can see "jne acrack.7FF706CC15BD". So, JNE means in assembly is **Jump if Not Equal**. We can see same for our Password.

So that's meaning is If our password doesn't match, Jump to "acrack.7FF706CC15BD". So, let's change the condition. What if we can change the **JNE** to **JE**? **JE** meaning is **Jump If Equal**. So will go to that line and press space button to assemble that line.

lea rdx,qword ptr ds:[7FF706CD1042]	00007FF706CD1042:"CrackMeOne"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call <JMP.&strcmp>	
test eax,eax	
je acrack.7FF706CC15BD	
lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D:"True!\n"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC14A4	
lea rax,qword ptr ds:[7FF706CD1054]	00007FF706CD1054:"Password: "
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC14A4	
mov rax,qword ptr ss:[rbp-10]	
mov rdx,rax	
lea rax,qword ptr ds:[7FF706CD103B]	00007FF706CD103B:"%s[^\n]"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC1450	
mov rax,qword ptr ss:[rbp-10]	
lea rdx,qword ptr ds:[7FF706CD105F]	00007FF706CD105F:"t.me/Umno223"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call <JMP.&strcmp>	
test eax,eax	
je acrack.7FF706CC15AC	
lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D:"True!\n"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC14A4	
jmp acrack.7FF706CC15CC	
lea rax,qword ptr ds:[7FF706CD106C]	00007FF706CD106C:"Bad password, try again\n"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC14A4	
jmp acrack.7FF706CC15CC	
lea rax,qword ptr ds:[7FF706CD1085]	00007FF706CD1085:"bad nickname! try again\n"
mov rcx,rax	rcx:ZwQueryInformationThread+14
call acrack.7FF706CC14A4	
mov eax,0	
add rsp,30	
pop rbp	
ret	



So, I just re assembled the **JNE** to **JE** both of lines. And now need to patch this application. So, I will press **CTRL+P** to patch the application. Then patch the application and save new application as **cracked.exe**. Let's execute out new program.



```
07FF706CC153D 48:8D15 FEFA0000 |lea rdx,qword ptr ds:[7FF706CD1042] 00007FF706CD1042: "CrackMeOne"
07FF706CC1544 48:89C1 |mov rcx,rcx rcx: ZwQueryInformationThread+14
C:\Windows\System32\cmd.exe
C:\Users\Piyusha Akash\Desktop\RevEng>dir *exe
Volume in drive C has no label.
Volume Serial Number is 26D1-2522

Directory of C:\Users\Piyusha Akash\Desktop\RevEng
12/15/2025 07:25 PM 383,745 acrack.exe
02/03/2026 06:27 PM 383,745 cracked.exe
2 File(s) 767,490 bytes
0 Dir(s) 38,245,040,128 bytes free

C:\Users\Piyusha Akash\Desktop\RevEng>cracked.exe
Mini crackme by @microsoftstorage written on C!
Nickname: .
True!
Password: .
True!

C:\Users\Piyusha Akash\Desktop\RevEng>
```

So, as you can see, we will successfully patch the application. Now it will work for any credentials. In that case I put "." As my Nickname and Password. And it's working now. So, what we did? What happened?

We will change the logic in assembly. In assembly this happened. If we entered wrong credentials, Logic jumps to bad credentials message. If it is correct, Logic goes normally. That's why we change **JNE** to **JE**.

Thank you for reading this writeup. Don't forget to follow me. Best regards, **Piyusha Akash. (0x3xp)**

LinkedIn: <https://linkedin.com/in/piyushaakash>

GitHub: <https://github.com/0x3xp>

YouTube: <https://youtube.com/@infosecl>

TikTok: <https://tiktok.com/@infosecl>

Linktree: <https://linktr.ee/piyushaakash>